International Core Journal of Engineering Volume 8 Issue 5, 2022
ISSN: 2414-1895 DOI: 10.6919/ICJE.202205_8(5).0023

Equilibrium-Aware Decentralized Cooperative Caching for Internet of Vehicles

Yizheng Fan, and Honghai Wu

Henan University of Science and Technology, Luoyang 471023, China

Abstract

Since edge caching is expected to solve a series of problems caused by limited spectrum and bandwidth resources, it has also become one of the most potential technologies to break through the bottleneck in the development of the Internet of Vehicles. However, many related studies are lacking in pertinence and difficult to adapt to the dynamic network environment targeting vehicles. Based on the dynamic cooperation among smart vehicles, Edge Base Stations (EBS) and Cloud Data Center (CDC), this paper proposes an equilibrium-aware decentralized cooperative caching for Internet of Vehicles by jointly optimizing caching node selection and content update. In this paper, we model the objective optimization problem as a Markov Decision Process (MDP), use the Deep Deterministic Policy Gradient algorithm to solve the formulated long-term mixed integer linear programming problem, and improve the federated learning training process by considering limited resources as well as vehicle privacy. Among them, an Attention-Weighted Decentralized Cooperative caching (AWDC) is proposed to optimize the caching model for different vehicles and dynamic environments. An attention mechanism is used to control the model weights in the federated learning aggregation step to address the imbalance in the quality of different local models. It assigns different aggregation weights to different quality models to ensure a more accurate model, which can effectively improve the hit rate, reduce the average delay and offload traffic, and improve the fairness among users.

Keywords

Internet of Vehicles; Cooperative Caching; Deep Reinforcement Learning; Federated Learning; Attention Mechanism.

1. Introduction

In recent years, with the rapid growth of vehicle ownership, the Internet of Vehicles has become a research and development hotspot due to its comprehensive advantages and huge potential. The popularization of Internet of Things and artificial intelligence technology has laid the foundation for the development of intelligent transportation systems. At the same time, the booming development of the automotive industry has not only realized the autonomous driving of vehicles, but also promoted many emerging applications and services, from user-defined in-vehicle infotainment to related applications to ensure vehicle safety. These applications and services may require a lot of computing, communication and storage resources, and have strict requirements on service response delay and network bandwidth[1]. Smart vehicles are equipped with on-board units that enable the vehicle to have computing and storage capabilities, and can communicate with roadside infrastructure, pedestrians, and other vehicles via C-V2X, LTE-V, or IEEE802.11p technologies[2]. Therefore, the ever-increasing computing-intensive and delay-sensitive vehicle applications have an urgent demand for intelligent transportation. Using roadside units and intelligent vehicles as caching nodes to establish a cooperative cache system can effectively solve the above problems.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202205_8(5).0023

The use of DRL in the work of [3] for dynamic cache resource management can significantly improve long-term utility performance. But vehicles don't have the extra performance to support large-scale data processing neural networks. Moreover, some of the existing work is limited by traditional DRL, and centralized learning algorithms are prone to waste bandwidth and computing resources on repetitive tasks, and consume a lot of network resources during data transmission [4]. Therefore, a multi-agent reinforcement learning-based collaborative caching strategy is considered in the work of [5], and [6] uses a distributed multi-agent Multi-Armed Bandit algorithm to maximize the accumulated cache under a certain time limit utility. In their work, it is unavoidable that their distributed learning algorithm requires a lot of cache resources and action space, and fails to solve the above problems well.

Federated learning, as a promising method in many fields, trains neural network parameters by saving the training data locally in the terminal [7]. There are also many studies on federated learning in mobile edge computing. For example, in the work of [8], the federated learning framework and DQN algorithm are jointly used to solve the related problem of computational task offloading. Meanwhile, joint federated learning framework and deep reinforcement learning can also solve the problem of joint computing, caching and communication in mobile edge networks [9]. However, none of them can avoid the problem of aggregation efficiency. In the face of different terminal performance, characteristics and different environments, the average aggregation cannot reasonably provide an effective global model.

Therefore, this paper explores the framework design of heterogeneous collaborative caching based on the Internet of Vehicles by jointly optimizing the caching node selection and content update in the Internet of Vehicles, and considers the dynamic tripartite cooperation between smart vehicles, EBS and cloud data centers. We formulate the objective optimization problem as MDP and propose an AWDC framework to solve the problem. Among them, DDPG is used to solve the formulated long-term mixed-integer linear programming problem, and federated learning is used to optimize the training process considering limited vehicle performance and user privacy. Most importantly, in the FL aggregation step, we utilize the attention mechanism to control the weights of different local models, avoiding the problem of ineffective aggregation due to the uneven quality of local models. The main contributions of this paper can be summarized as:

1) This paper studies the problem of decentralized edge cooperative caching in social car networking. We model the caching process by constructing a long-term mixed-integer linear programming problem, and use the DDPG algorithm to dynamically control the selection of caching nodes and the decision-making process of cache update based on the state of the network and the vehicle itself, as well as the historical data of the vehicle.

2) We propose an attention-weighted decentralized edge cooperative caching strategy, an improved federated learning framework through the attention mechanism, save the training data locally in the vehicle and train the DDPG model in a decentralized manner, which solves the problem between heterogeneous vehicles. Model aggregation problem. The key step is that in the stage of global aggregation of local models, we use the attention mechanism to control the weights of different local models to avoid invalid aggregation due to the unbalanced quality of local models.

3) Simulation results show that, compared with existing methods, our proposed attention-weighted decentralized edge cooperative caching strategy can greatly reduce average latency, improve hit rate and fairness among users, and reduce system load and overhead.

2. System Model

In this section, we introduce a decentralized heterogeneous cooperative caching system model in the Internet of Vehicles. It mainly starts from the corresponding network architecture, modeling of content popularity and user preferences, vehicle sharing assistance mode, and content transmission and delay models.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202205_8(5).0023

2.1 Network Architecture

We consider a multi-layer urban transportation network architecture based on social vehicle networking, as shown in Figure 1, which includes three types of heterogeneous caching nodes: CDC, EBS, and smart vehicles. We assume that CDC stores all available contents in the content library, the index set of available contents in the network is $\mathcal{K} = \{1,2,\dots,\mathrm{K}\}$, and its data size is represented by $(D_k)^{1\times \mathrm{K}}$.At the same time, $\mathcal{V} = \{1,2,\dots,N\}$ is used to represent the index set of all smart vehicles, and each vehicle has a limited local cache resource C_n. In order to make our system easier to implement, we only consider the cooperation scenarios of two adjacent EBSs in this model, namely the EBS (denoted by EBS^s) that serves the vehicle and the adjacent EBS that is closest to [EBS] and maintains communication (denoted by EBS^c), both EBSs are placed with edge servers and have limited cache resources C_B. In addition, optical fiber communication is used between EBSs, and the CDC sends the cached content to the EBSs through the backhaul link. We divide the considered time period into T segments and denote the time slot as $\tau = \{0,1,2,\dots,\mathrm{T}\}$.

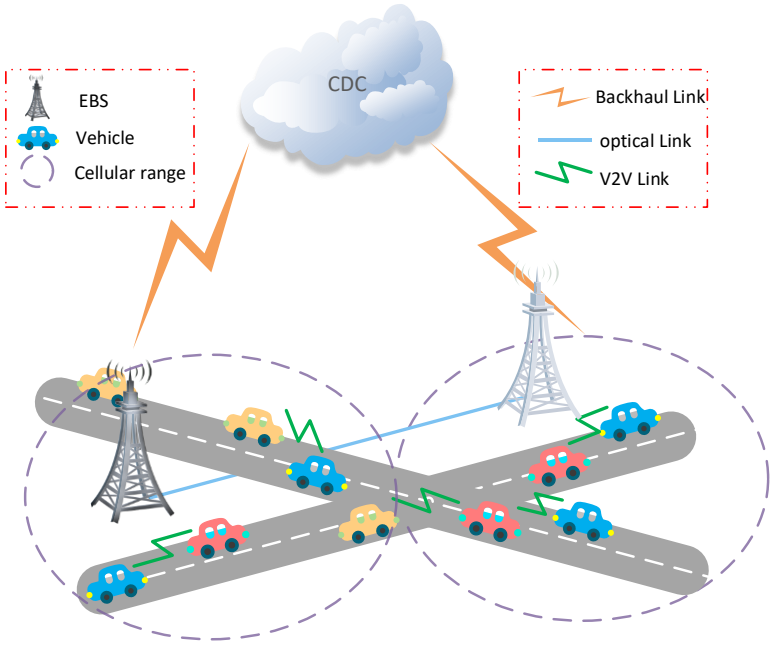


Figure 1. Network architecture of heterogeneous collaborative cache in Internet of Vehicles.

We assume that the global content popularity obeys the Zipf distribution[10] and is expressed as:

$$\mathcal{P}_k = \frac{(R_k+\beta)^{-\alpha}}{\sum_{i\in\mathcal{F}}(R_i+\beta)^{-\alpha}}, \qquad \forall k \in \mathcal{K}. \tag{1}$$

where R_k is the popularity ranking of content k in the content library, and α and β are the skewness factor and platform factor, respectively. Also, denote the set of all content popularity by $\mathbb{p} = (\mathcal{P}_k)^{1\times \mathrm{K}}$, and we assume that the global content popularity changes smoothly over a long period of time.

We assume that the vehicle's preference for content is $\mathcal{p}_n^k$, which represents the vehicle's access probability to content k:

$$\mathcal{p}_n^k = \frac{q_{n,k}}{q_n}, \qquad \forall n \in \mathcal{V}, \forall k \in \mathcal{K}. \tag{2}$$

ISSN: 2414-1895 DOI: 10.6919/ICJE.202205_8(5).0023

Where q_n is the total number of requests initiated by vehicle n for all content in this time period, and $q_{n,k}$ is the number of requests for content k by vehicle n in this time period. Satisfy $\sum_{k\in\mathcal{K}} p_n^k = 1$ for any vehicle n.

2.2 V2V Collaborative Model

Smart vehicles running on the road network equipped with cache resources can act as content carriers, store requested popular content locally in the vehicle, and forward the cached content to nearby vehicles through V2V communication. When the distance between adjacent vehicles and the network status meets the conditions for establishing a communication link, and the supply and demand content between the two are consistent, we consider them to meet the requirements of social relations. In order to judge whether the requirements are met between vehicles, on the one hand, the content of supply and demand between vehicles is matched. We use the content preference p_n^k of the vehicle to represent the cached probability of the content k of the vehicle n, and the probability set of the vehicle n to cache all the content is $\boldsymbol{P}_n = \{p_n^1, p_n^2, \dots, p_n^K\}$, that is, the probability that the target vehicle encounters the target content k cached by vehicle n on the road is p_n^k, so the matching probability of the content can also be used $\boldsymbol{P}_n$ Express. On the other hand, we introduce an exponentially distributed vehicle contact rate ε_n [11], from which we can obtain the probability of V2V encounter and the V2V encounter time. Therefore, we can calculate the communication probability of V2V between vehicles n and m $(m \in \boldsymbol{\mathcal{V}})$ as $g_m^n = p_n^k \varepsilon_n$ and denote it as $\boldsymbol{G}^n = (g_m^n)^{1\times N}$. And we normalize $\boldsymbol{G}^n$ by $\frac{g_m^n}{\sum_{m\in\mathcal{V}} g_m^n} \to g_m^n$ for any $n, m \in \boldsymbol{\mathcal{V}}$.

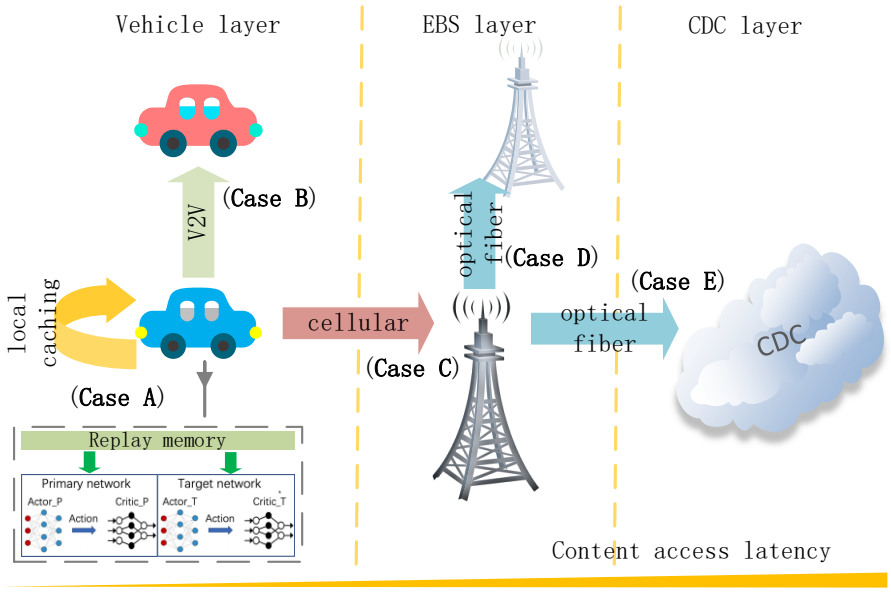


Figure 2. Several ways for vehicles to obtain the requested content.

2.3 Transmission and Delay Models

In each time slot $t \in \tau$, the smart vehicle will send a download request Req_n^t for the required content, and use $\boldsymbol{\mathcal{R}}_t = \{Req_1^t, Req_2^t, \dots, Req_n^t\}$ to describe the system request state at time slot t, the requested content at this time is marked as $k_{n,t}$. As shown in Figure 2, we enumerate the scenarios in which all vehicles can obtain the requested content:

Case A: If the content of the vehicle n has been cached locally, the vehicle does not need to initiate a request to other caching nodes, and directly obtains the requested content locally. We use $d_{n,t}^L$ to represent the delay of obtaining the requested content from the local, which is so small that it can be ignored.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202205_8(5).0023

Case B: When vehicle n does not cache the content locally, it will initiate a request to the surrounding cooperative vehicles, and establish V2V communication with vehicles that have cached the content. Communication between vehicles uses dedicated short-range communication technology based on 802.11p, and the V2V data transfer rate is denoted by r^V [12]. Therefore, the transmission delay required for vehicle n to obtain content k from candidate vehicle at time slot t is $d_{n,t}^V = \frac{D_{k_{n,t}}}{r_t^v}$, where $D_{k_{n,t}}$ represents the data amount of the content $k_{n,t}$.

Case C: Vehicles can also obtain the required content from the cache library of EBS^s. From [106], the data transfer rate of the vehicle and EBS^s can be obtained as $r_{n,t}^E = W \log_2\left(1 + \frac{P l_n^{-\theta} |h|^2}{\varphi}\right)$, where W is the channel bandwidth occupied by the vehicle, l_n is the distance between vehicle n and EBS^s, θ is the path loss index, φ is the power of white Gaussian noise, and P is the wisdom The transmission power of the vehicle, h is the channel attenuation coefficient. From this, it can be obtained that the delay required for the vehicle to obtain the content at this time is $d_{n,t}^E = \frac{D_{k_{n,t}}}{r_{n,t}^E}$.

Case D: In the case that the content requested by the vehicle is not cached by EBS^s, it can be considered to obtain the content from the EBS^C that has cached the content. At this time, we also need to consider the transmission delay between EBSs, and use stable optical fibers to transmit data between them, and the average data transmission rate is r^C. From this, we can obtain that the delay required to obtain the requested content through the cooperation of EBSs is $d_{n,t}^C = d_{n,t}^E + \frac{D_{k_{n,t}}}{r^C}$.

Case E: If the required content is not found in the above methods, EBS^s will forward the request of the vehicle to CDC, and then download the data from CDC to the vehicle through EBS^s. At this time, we also need to consider the delay from CDC to EBS^s, which also uses optical fiber transmission, so it can be expressed by the average data transfer rate r^{CD}. From this, we can obtain that the delay required for the vehicle to obtain the content from the CDC is $d_{n,t}^{CD} = d_{n,t}^E + \frac{D_{k_{n,t}}}{r^{CD}}$.

2.4 Problem Formulation

In this section, we still choose to use MDP to model the problem of candidate vehicle selection and node cache update. In the following, we mainly describe the states and actions of intelligent vehicles and the calculation of system rewards in detail. The long-term goal of our vehicle is to find the optimal cached decision to optimize the desired long-term reward, and define this goal as a value function to solve. All caching nodes set the cache status of a content as follows: For all vehicles, if vehicle n has cached content k locally, it is expressed as $f_k^n = 1$, otherwise it is equal to 0; for EBS^s and EBS^C, if they have been cached locally The content k is expressed as $f_k^S = 1$ and $f_k^C = 1$ respectively, otherwise it is expressed as $f_k^S = 0$ and $f_k^C = 0$. Therefore, we denote the caching states of vehicle n, EBS^s and EBS^C for all contents as $\boldsymbol{F}_t^n$, $\boldsymbol{F}_t^S$ and $\boldsymbol{F}_t^C$, respectively.

2.4.1 Vehicle State

We express the state of vehicle n at time slot t as $\boldsymbol{S}_t^n = \{\mathbb{p}_t, \boldsymbol{\mathcal{P}}_n, r_t^v, \boldsymbol{F}_t^n, \boldsymbol{O}_k\}$, where $\mathbb{p}_t$ and $\boldsymbol{\mathcal{P}}_n$ are the popularity of all content and vehicle at time slot t, respectively user preference of n, and $\boldsymbol{O}_k$ is an indicator of the cache location of content k.

2.4.2 Vehicle Action

After receiving the state $\boldsymbol{S}_t^n$, vehicle n will select a method to obtain the content according to the above method, and replace the content that needs to be updated in the local cache list. The action of vehicle n at time slot t can be expressed as $\boldsymbol{A}_t^n = \{\psi_{i,t}^{n,k}, \boldsymbol{F}_t^{n,k}\}$, where $\psi_{i,t}^{n,k}$ An indicator indicating whether vehicle n obtains the requested content k from caching node i. Caching nodes include nearby cooperative vehicles, EBS and CDC. $\boldsymbol{F}_t^{n,k}$ indicates whether the content k of the local cache of vehicle n needs to be replaced.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202205_8(5).0023

2.4.3 System Reward

Our goal is to maximize the amount of content data assisted by V2V communication, while also ensuring that the delay in vehicle acquisition of content is minimized. Therefore, our setting of the reward function includes two parts: the gain function of V2V assisting cache and the gain of content access delay. For each pair of smart vehicles at time slot t, only case B can generate a V2V cooperative gain, that is, vehicle n has no local cache content k ($f_t^n = 0$), but vehicle m has a cache ($f_t^m = 1$), the probability that vehicle n can obtain content k from vehicle m is g_m^n. Therefore, at time slot t, the V2V cooperation gain of vehicle n and vehicle m can be calculated as $D_k g_m^n$, then the total gain of V2V cooperative cache delivery is:

$$\mathcal{R}_t^{1,n} = \sum_{\substack{m\in\boldsymbol{\mathcal{V}} \\ k\in\boldsymbol{\mathcal{K}}}} D_k g_m^n f_t^m (1 - f_t^n), \qquad \forall n \in \boldsymbol{\mathcal{V}}, \forall t \in \tau. \tag{3}$$

Normally, $d_{n,t}^{CD} < d_{n,t}^{C} < d_{n,t}^{S} < d_{n,t}^{V} < d_{n,t}^{L}$ at the edge established in the caching system [13]. We normalize the reward function using the probability density function of a negative exponential distribution and define the gain in content access latency as:

$$\mathcal{R}_t^{2,n} = \begin{cases} \mathcal{P}_k e^{-d_{n,t}^{L}}, & Case\ A \\ \mathcal{P}_k e^{-d_{n,t}^{V}}, & Case\ B \\ \mathcal{P}_k e^{-d_{n,t}^{S}}, & Case\ C \\ \mathcal{P}_k e^{-d_{n,t}^{C}}, & Case\ D \\ \mathcal{P}_k e^{-d_{n,t}^{CD}}, & Case\ E \end{cases}. \tag{4}$$

Based on the above two gains, the reward function of the cache system is defined as:

$$\mathcal{R}(\boldsymbol{S}_t^n, \boldsymbol{A}_t^n) = \zeta^1 \mathcal{R}_t^{1,n} + \zeta^2 \mathcal{R}_t^{2,n}, \qquad \forall n \in \boldsymbol{\mathcal{V}}, \forall t \in \tau. \tag{5}$$

We introduce weighting factors ζ^1 and ζ^2, respectively, and $\zeta^1 + \zeta^2 = 1, 0 \le \zeta^1, \zeta^2 \le 1$.

2.4.4 Target Value Function

We define the mapping from the current state to the next decision-making action as the cache policy $\mathbb{A}$, that is, when the cache policy $\mathbb{a}$ is executed in the state $\boldsymbol{S}_t^n$, the probability of taking the next action $\boldsymbol{a}$ is $\mathbb{a}(\boldsymbol{a}|\boldsymbol{S}_t^n)$, and the action $\boldsymbol{A}_t^n$ can be obtained by $\boldsymbol{A}_t^n = \underset{\boldsymbol{A}}{\operatorname{argmax}}\, \mathbb{a}(\boldsymbol{a}|\boldsymbol{S}_t^n)$. At this point, the goal of the intelligent vehicle can be transformed into finding an optimal cache decision to continuously optimize the expected reward value function:

$$V(\boldsymbol{s}) = E_{\boldsymbol{A}}\left[\sum_{t=0}^{\infty} \eta^{t-1} \cdot \mathcal{R}(\boldsymbol{S}_t^n, \boldsymbol{A}_t^n) | \boldsymbol{S}_0^n = \boldsymbol{s}\right], \tag{6}$$

where $\eta \in [0,1]$ is the discount coefficient. Then through the Bellman equation, the value function in (6) is expressed as:

ISSN: 2414-1895 DOI: 10.6919/ICJE.202205_8(5).0023

$$V(\boldsymbol{s}) = \sum_{\boldsymbol{a}\in\boldsymbol{A}} \mathbb{a}(\boldsymbol{a}|\boldsymbol{s})\left\{\mathcal{R}(\boldsymbol{s},\boldsymbol{a}) + \eta \sum_{\boldsymbol{s'}\in\boldsymbol{A}} \Pr\{\boldsymbol{s'}|(\boldsymbol{s},\boldsymbol{a})\}\cdot V(\boldsymbol{s'})\right\}, \tag{7}$$

Our goal is to minimize the content access latency of the system, so it is expected that each vehicle can learn and obtain the optimal caching policy $\mathbb{a}^* \in \mathbb{A}$. We formulate the problem of multi-layer node selection and cache content update for decentralized cooperative caching in the Internet of Vehicles as the maximizing objective value function:

$$\begin{aligned} &\max_{\mathbb{a}^*\in\mathbb{A}} V(\boldsymbol{s}) \\ &\boldsymbol{s.t.} \quad C1:\ f_k^n \in \{0,1\}, \qquad n \in \boldsymbol{\mathcal{V}}, \forall k \in \mathcal{K} \\ &\qquad\quad C2:\ \sum_{k\in\mathcal{K}} D_k f_k^n \le C_n, \qquad \forall n \in \boldsymbol{\mathcal{V}} \quad . \end{aligned} \tag{8}$$

The restriction is to ensure that the cached content does not exceed the vehicle's cache space. Since the above problem is a long-term mixed integer linear programming problem, it has been proved to be an NP-hard problem in [14]. Also, the dimensionality of the state space grows with the number of vehicles. When the number of vehicles and content is large enough, traditional iterative methods are limited to commonly used caching systems due to their extremely high computational complexity [15]. Therefore, we utilize the DDPG learning algorithm based on the federated learning framework to solve this problem.

3. Design of AWDC Framework

In order to solve these problems in the Internet of Vehicles scenario, we propose an AWDC framework, which is mainly divided into three stages: global model release, local model training, and attention-weighted aggregation. We will introduce the complete flow in the AWDC framework and describe the three stages in detail in turn.

3.1 Global Model Release

Before each round of training starts, each smart vehicle will report the real-time status to the EBS that currently establishes communication to determine whether to participate in a new round of training. After receiving the status information of vehicles within the service range, EBS publishes the global model obtained from the previous round of training to the smart vehicles participating in the new round of training, and starts a new round of local model training locally on the vehicle.

3.2 Local Model Training

In the local model training phase, after each smart vehicle participating in a new round of training receives the global model parameter Π_t from EBS, it will start training its local DDPG model π_t^n according to this parameter Π_t and local data. The local DDPG model on the vehicle side evaluates the policy based on the action value function $Q(\boldsymbol{S}_t^n, \boldsymbol{A}_t^n)$. According to its relationship with the value function, we can deduce:

$$Q(\boldsymbol{S}_t^n, \boldsymbol{A}_t^n) = \mathcal{R}(\boldsymbol{S}_t^n, \boldsymbol{A}_t^n) + \eta \sum_{\boldsymbol{S}_{t+1}^n\in\boldsymbol{A}} \Pr\{\boldsymbol{S}_{t+1}^n|(\boldsymbol{S}_t^n, \boldsymbol{A}_t^n)\}\cdot V(\boldsymbol{S}_{t+1}^n), \tag{9}$$

Since $Q(\boldsymbol{S}_t^n, \boldsymbol{A}_t^n)$ is a continuous action space that is difficult to obtain directly, we use the DDPG algorithm to solve this problem. The DDPG algorithm has two parts, the Actor network and the Critic network, and each network is composed of an estimation network and a target network.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202205_8(5).0023

Actor estimation network: At this time, the state $\boldsymbol{S}_t^n$ is used as the input, and then the probability of different actions is obtained according to the exploration strategy π^A, and the most suitable action $\boldsymbol{A}_t^n$ in this round is selected among these continuous actions. The actor estimation network uses the sampled policy gradient to update the local parameter π^n, expressed as:

$$\nabla\pi^A = \frac{1}{N}\sum\nolimits_i \nabla_{\boldsymbol{a}} Q(\boldsymbol{S}_i^n, \boldsymbol{A}_i^n|\pi^C)\nabla_{\pi^A}\mathbb{a}(\boldsymbol{S}_i^n|\pi^A), \tag{10}$$

Where $Q(\boldsymbol{S}_i^n, \boldsymbol{A}_i^n|\pi^C)$ is the action value function, and $\mathbb{a}(\boldsymbol{S}_i^n|\pi^A)$ is the caching strategy.

Critic estimation network: This part is to evaluate the performance of the selected actions and update the local model parameters π^C according to the action value function and the minimum loss function. The action value function of the Critic estimation network is:

$$Q(\boldsymbol{S}_i^n, \boldsymbol{A}_i^n|\pi^C) = \frac{1}{N}\sum\nolimits_i \mathcal{R}(\boldsymbol{S}_i^n, \boldsymbol{A}_i^n) + \in Q(\boldsymbol{S}_{i+1}^n, \mathbb{a}(\boldsymbol{S}_{i+1}^n)|\pi^C), \tag{11}$$

where $\in$ denotes the probability of choosing a random action. Meanwhile, the loss function is expressed as:

$$L(\pi^C) = \frac{1}{N}\sum\nolimits_i \left(\mathcal{R}(\boldsymbol{S}_i^n, \boldsymbol{A}_i^n) + \eta Q'\left(\boldsymbol{S}_{i+1}^n, \mathbb{a}'(\boldsymbol{S}_{i+1}^n)|\pi^{C'}\right) - Q(\boldsymbol{S}_i^n, \boldsymbol{A}_i^n|\pi^C)\right)^2, \tag{12}$$

Target network: The target network structure in Actor network and Critic network is the same, but the parameters are different. Every so often, the estimation network copies the local parameters $\tilde{\pi}$ to the target network, expressed as:

$$\tilde{\pi} = \sigma\pi + (1-\sigma)\tilde{\pi}, \qquad \sigma \in [1, 0]. \tag{13}$$

Among them, the learning coefficient σ is a constant, and its value is usually close to 1.

We will store the single batch training set ($\boldsymbol{S}_t^n, \boldsymbol{A}_t^n, \boldsymbol{\mathcal{R}}_t^n, \boldsymbol{S}_{t+1}^n$) obtained by the vehicle interacting with each time slot in the experience replay memory $\mathcal{M}$. Then the estimation network and the target network randomly select these training sets for training, which avoids the result that the continuous extraction of relevant training sets affects the results of local parameters.

3.3 Attention-weighted Aggregation

Typically, federated learning frameworks use an average method of aggregation, which is based on a simple weighted average integration based on the data size of the distributed training local model parameters [16]. However, we consider the quality difference of different local models, and it is unreasonable to simply aggregate them on average. In our model, we use weighted federated aggregation, as shown in Figure 3. Mainly consider the reward value $\mathcal{R}(\boldsymbol{S}_t^n, \boldsymbol{A}_t^n)$ and some performance indicators of the vehicle itself as metrics to evaluate the contribution of each local model to the global model. At this point, the related problem of weighted federated aggregation can be formulated as:

$$\min_{\pi,\varpi}\left\{F(\Pi_t) \triangleq \sum\nolimits_{n\in\boldsymbol{\mathcal{V}}} \theta^n F_n(\pi_t^n)\right\}, \tag{14}$$

ISSN: 2414-1895 DOI: 10.6919/ICJE.202205_8(5).0023

where θ^n is introduced as the weight factor of the local model of vehicle n to measure the contribution to the global model.

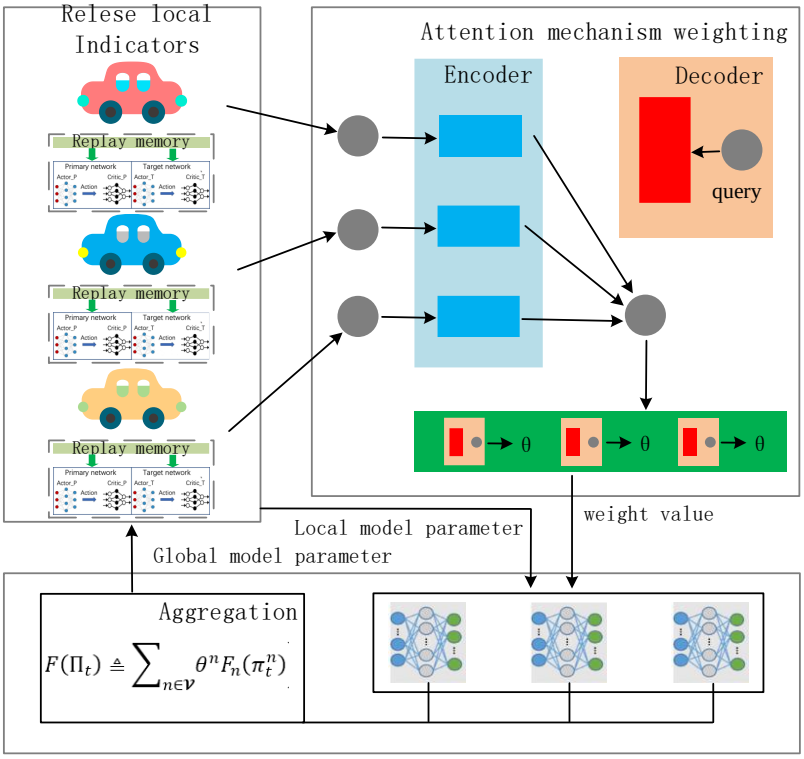


Figure 3. Federated aggregation process based on attention mechanism.

In this chapter, the weight θ^n of vehicle n is calculated including its average reward value $\bar{\mathcal{R}}^n$, average loss value $\bar{L}^n$, training data volume $\mathcal{M}^n$, and the number of datasets per training $\widetilde{\mathcal{M}}^n$, the number of local model training $\mathcal{E}^n$ and the average caching hit rate $\bar{H}^n$. The average reward value $\bar{\mathcal{R}}^n$ of vehicle n is the average of all local rewards $\mathcal{R}(\boldsymbol{S}_i^n, \boldsymbol{A}_i^n)$ obtained by $\mathcal{E}^n$ local training. The average loss value $\bar{L}^n$ for vehicle n is calculated by averaging all loss functions $L(\pi_i^C)$ output in $\mathcal{E}^n$ local trainings. The amount of training data for vehicle n is the size $\mathcal{M}^n$ of the experience playback memory. The more playback memory, the more historical training data can be stored locally. The number of datasets for each training of vehicle n is the size of the mini-batch memory $\widetilde{\mathcal{M}}^n$, and more data can be trained each time to obtain a more accurate local model. We describe these metrics as $\mathbb{K}^n = [\bar{\mathcal{R}}^n, \bar{L}^n, \mathcal{M}^n, \widetilde{\mathcal{M}}^n, \mathcal{E}^n, \bar{H}^n]$. With the maturity of deep network research, attention mechanism is also widely used in many fields[17]. In this chapter's decentralized model, we employ an attention mechanism for weighted federated aggregation. We model the evaluation index vector $\mathbb{K}^n$ and the local model parameters trained by vehicle n as key and value terms in the attention mechanism, respectively. The goal of our model is to get a better performing agent for more reward, lower loss, and higher hit rate, so we denote the query term as:

$$\mathbb{Q} = \left[\max_n \bar{\mathcal{R}}^n, \min_n \bar{L}^n, \max_n \mathcal{M}^n, \max_n \widetilde{\mathcal{M}}^n, \max_n \mathcal{E}^n, \max_n \bar{H}^n\right], \tag{15}$$

Therefore, the input of the EBS side includes the local model parameter π_t^n, the query item $\mathbb{Q}$ and the dimension ℓ of the key item $\mathbb{K}^n$. Then calculate the dot product of the query item and all key

ISSN: 2414-1895 DOI: 10.6919/ICJE.202205_8(5).0023

items, divide them by $\sqrt{\ell}$, and finally use the softmax function to obtain the weight of the local model. For any vehicle n, the expression of the weighting factor θ^n is as follows:

$$\theta^n = Attention(\mathbb{Q}, \mathbb{K}^n) = softmax\left(\frac{\mathbb{Q}(\mathbb{K}^n)^T}{\sqrt{\ell}}\right), \qquad \forall n \in \boldsymbol{\mathcal{V}}. \tag{16}$$

4. Simulation Results

4.1 Parameter Setting

Let's assume the number of vehicles is from 50 to 400. CDC stores all available content, and the cache sizes for vehicles and EBS are set to 300MB and 2GB by default. The number of contents is 10000 by default, and the size range of each data size is set from 5MB to 10MB. To compare vehicle computing instances and data sizes in different scenarios, we deploy different experience replay memories in vehicles, ranging in size from 1000 to 2000, mini-batch memory size is set from 128 to 256, and aggregation step size is set to 100. In each round of global training, it is assumed that the number of local model training times $\mathcal{E}^n$ ranges from 100 to 200, and the learning coefficient σ is set to 0.05 and the reward decay η to 0.9. Furthermore, the channel gain is modeled as $g_{m,n} = 36.8 + 36.7 \log_{10} l_{m,n}$dB, the channel bandwidth is 20MHz, and the transmit power is 40W.

4.2 Evaluate Results

In this section, in order to evaluate the performance of the AWDC algorithm proposed in this chapter in different scenarios, we evaluate the caching system from three performances: Average content Access Latency (AAL), Caching Hit Rate (CHR) and Traffic Offload Ratio (TOR). evaluate. And the following benchmark schemes are introduced for comparison:

1) LRU: On each update, replace the content that hasn't been requested for the longest time.

2) LFU: On each update, replace the content that was requested the least number of times.

3) DMARL[5]: Considering the influence of multi-agent on the cache environment, a distributed multi-agent deep reinforcement learning caching strategy is established based on the Markov game model.

4) FedAvg[18]: A DDPG model is run in each vehicle, and after local training is complete, the local models are aggregated on average according to the data size.

5) FedProx错误!未找到引用源。: Generalization and parameter redefinition of FedAvg, which improves the aggregation of local models according to changes in network features.

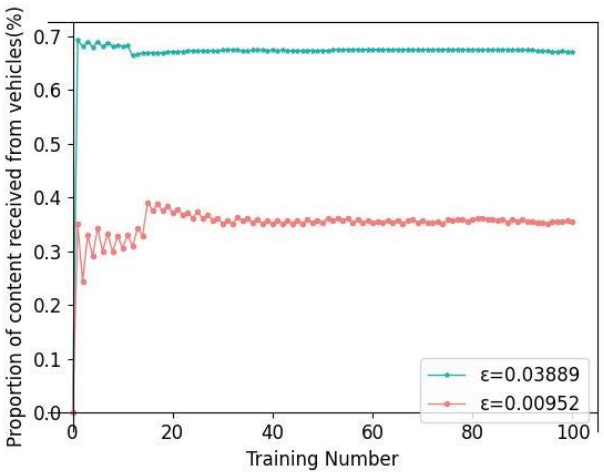


Figure 4. Convergence of the AWDC algorithm.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202205_8(5).0023

We consider user preferences to be random and the number of vehicles involved in training is 400. Figure 4 shows the convergence of the AWDC algorithm under the conditions of different vehicle contact rates, where the vehicle contact rates are 0.03889 and 0.00952, respectively. In environments with different vehicle social intensities, the algorithm starts to converge around 15 iterations.

Based on the attention mechanism introduced in this paper, we can get the attention weights of different vehicles when globally aggregated. The change of the attention weight of each intelligent vehicle participating in the training is shown in Figure 5, where the fluctuation of the weight value is due to the change of the training data size. We can clearly see that the vehicle 4 with the largest weight value accounts for about 29%, while the vehicle 1 with the smallest weight value accounts for about 22%. The decisive factor for this situation is that the vehicle 4 has stronger performance. The area of each color in the figure also determines the influence of the corresponding vehicle on the global model. Obviously, the area of vehicle 4 is larger than that of vehicle 1.

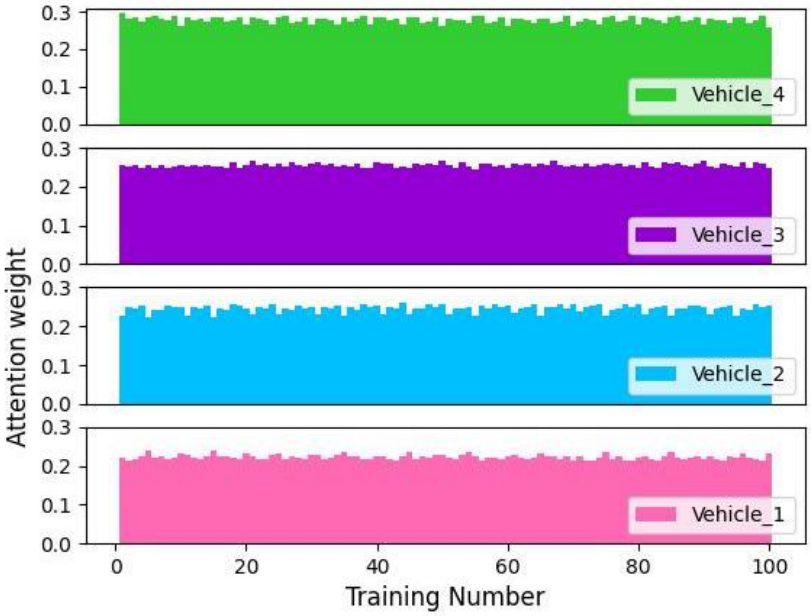
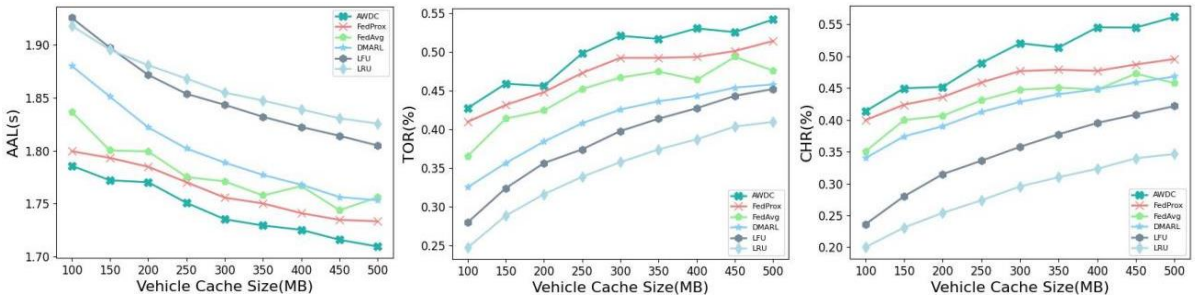


Figure 5. Changes of different vehicles' attention weights.

Figure 6 shows the performance comparison of different schemes under different vehicle cache sizes. Since our proposed AWDC algorithm can more intelligently follow user preferences and content popularity to select cached content, it can better adapt to users' random preferences. In Figure 6(a), we can clearly see that the algorithm proposed in this paper can reduce the delay ranging from 45ms to 85ms compared to other algorithms. In Figure 6(b), the AWDC algorithm is able to offload 8% to 21% more backhaul links than other algorithms. In Figure 6(c), the AWDC algorithm can also improve the cache hit rate by about 7% to 19%. Among the compared learning algorithms, the algorithm based on the federated learning framework is obviously better than the general distributed learning algorithm, and our AWDC algorithm is also optimal in various performance indicators. With the improvement of vehicle cache resources, there will be more information about the input state of the local DDPG, and it will become more and more difficult to obtain the global model by simply aggregating the local models on average, such as FedAvg. The FedProx and AWDC algorithms have one thing in common is that they both consider the state of the local model, which allows different agents to be trained locally for different times. The above proves that our proposed AWDC algorithm can effectively select caching nodes according to the network state.

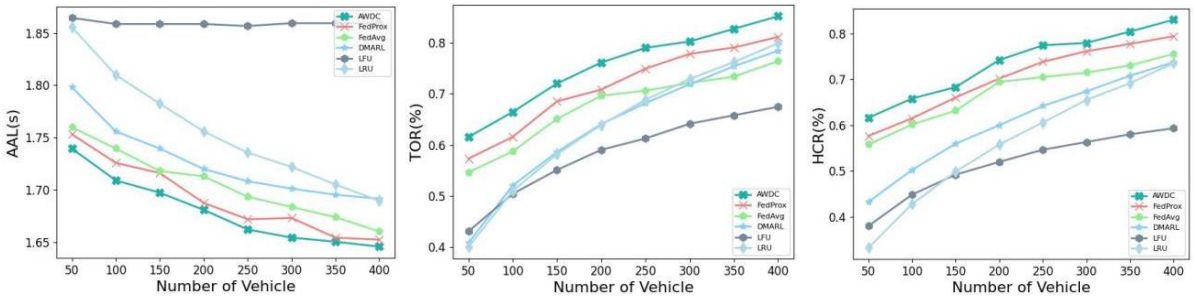
ISSN: 2414-1895 DOI: 10.6919/ICJE.202205_8(5).0023



(a)AAL. (b)TOR. (c)CHR.

Figure 6. Performance comparison of different schemes under different vehicle cache size.

Figure 7 shows the performance comparison of several different schemes with different numbers of participating training vehicles. We choose the number of vehicles in the range of 50 to 400, limited by the number of terminals that the cellular network can serve simultaneously. As can be seen from Figure 7(a), the AWDC algorithm proposed in this paper is compared with all the reference algorithms, and it is obvious that the average content access delay of the AWDC algorithm is the lowest. It can be seen from Figure 7(b) that the traffic offloaded by the backhaul link increases with the number of vehicles, which means that when the edge node resources are sufficient, the target vehicle will select the edge node with a closer distance to obtain the request content. And we can see that in the case of abundant edge resources, the AWDC algorithm has the least dependence on the backhaul link. As can be seen from Figure 7(c), AWDC still outperforms other reference algorithms in terms of cache hit rate, and can improve them by 9% to 21%. In addition, in all performance metrics comparisons, the performance gap between the AWDC algorithm and FedAvg also becomes larger as the number of participating training vehicles increases. Because there are more vehicles, there are more local models with uneven quality, and simple average aggregation cannot get a global model with accurate performance.



(a)AAL. (b)TOR. (c)CHR.

Figure 7. Performance comparison of different schemes under different vehicle cache size.

To sum up, the cooperative caching strategy based on the AWDC algorithm proposed in this paper can not only reduce the system overhead by reducing the load on the backhaul link, but also ensure a better average cache hit rate with a lower average delay. In particular, our proposed AWDC algorithm always outperforms the reference learning algorithm in different situations, which fully verifies the superiority of the AWDC strategy.

5. Summary

In this article, we have investigated the issue of decentralized cooperative caching and unequal model aggregation, and propose Equilibrium-Aware Decentralized Cooperative Caching for Internet of Vehicles. Among them, the deep reinforcement learning algorithm is used to dynamically make caching decisions based on vehicle performance and local historical data. In order to solve the above

ISSN: 2414-1895 DOI: 10.6919/ICJE.202205_8(5).0023

problems, we propose the AWDC framework to train a DDPG model in a decentralized way based on federated learning, and save the training data locally in the vehicle, which fully solves the problem of dealing with continuous action spaces and problems with model aggregation of heterogeneous vehicles. The attention mechanism is used to control the weights of different local models, which avoids the problem of invalid aggregation due to unbalanced quality of local models. The simulation results show that the proposed AWDC strategy is better than the comparison caching schemes, which can effectively improve the performance of the caching system, and it is proved that the introduction of the attention mechanism into the federated learning framework can effectively improve the average reward of the system.

References

[1] Liu L, Chen C, Pei Q, et al. Vehicular Edge Computing and Networking: A Survey. Mobile Networks and Applications. Vol. 16 (2020), p.1145–1168.

[2] Li FW, Zhang HB, Wang ZX. V2X Collaborative Caching and Resource Allocation in MEC-Based IoV. Journal on Communications. Vol. 42 (2021) No. 2, P. 26-37.

[3] Chen X, Zhang H, Wu C, et al. Optimized Computation Offloading Performance in Virtual Edge Computing Systems Via Deep Reinforcement Learning. IEEE Internet of Things Journal. Vol. 6 (2019) No. 3, P. 4005-4018.

[4] Lin P, Song Q, Song J, et al. Cooperative Caching and Transmission in CoMP-Integrated Cellular Networks Using Reinforcement Learning. IEEE Transactions on Vehicular Technology. Vol. 69 (2020) No. 5, P. 5508-5520.

[5] Jiang K, Zhou H, Zeng D, et al. Multi-Agent Reinforcement Learning for Cooperative Edge Caching in Internet of Vehicles. 2020 IEEE 17th International Conference on Mobile Ad Hoc and Sensor Systems (MASS). Online, 2020, P. 455-463.

[6] Xu X, Gao S, Tao M. Distributed Online Caching for High-Definition Maps in Autonomous Driving Systems. IEEE Wireless Communications Letters. Vol. 10 (2020) No. 7, P. 1390-1394.

[7] Li T, Sahu AK, Talwalkar A, et al. Federated Learning: Challenges, Methods, and Future Directions. IEEE Signal Processing Magazine. Vol. 37 (2020) No. 3, P. 50-60.

[8] Ren J, Wang H, Hou T, Zheng S, Federated Learning-Based Computation Offloading Optimization in Edge Computing-Supported Internet of Things. IEEE Access. Vol. 7 (2019), P. 691979201.

[9] Wang X, Han Y, Wang C, et al. In-Edge AI: Intelligentizing Mobile Edge Computing, Caching and Communication by Federated Learning. IEEE Network. Vol. 33 (2019) No. 5, P. 156-165.

[10] Breslau L, Pei C, Li F, et al. Web caching and Zipf-like distributions: evidence and implications. IEEE INFOCOM '99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No.99CH36320). New York, 1999, P. 126-1.

[11] Gao W, Li Q, Zhao B, et al. Social-Aware Multicast in Disruption-Tolerant Networks. IEEE/ACM Transactions on Networking. Vol. 20 (2012) No. 5, P. 1553-1566.

[12] Luan TH, Shen X, Bai F. Integrity-oriented content transmission in highway vehicular ad hoc networks. 2013 Proceedings IEEE INFOCOM. Turin, 2013, p. 2562–2570.

[13] Wang XF, Wang CY, Li XH, et al. Federated deep reinforcement learning for Internet of Things with decentralized cooperative edge caching. IEEE Internet of Things Journal, Vol. 7 (2020) No. 10, p. 9441-9455.

[14] Liu L, Chen C, Pei Q, et al. Vehicular Edge Computing and Networking: A Survey. Mobile Networks and Applications. Vol. 16 (2020), p. 1145–1168.

[15] Qiao G, Leng S, Maharjan S, et al. Deep reinforcement learning for cooperative content caching in vehicular edge computing and networks. IEEE Internet Things J. Vol. 7 (2020) No. 1, p. 247–257.

[16] Lim WYB, Luong NC, Hoang DT, et al. Federated Learning in Mobile Edge Networks: A Comprehensive Survey. IEEE Communications Surveys & Tutorials, Vol. 22 (2020) No. 3, p. 2031-2063.

[17] Zn A, Gz A, Hui YB. A Review on the Attention Mechanism of Deep Learning. Neurocomputing, Vol. 452 (2021) No. 10, p. 48-62.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202205_8(5).0023

[18] Mills J, Hu J, Min G. Communication-Efficient Federated Learning for Wireless Edge Intelligence in IoT. IEEE Internet of Things Journal, Vol. 7 (2020) No. 7, p. 5986-5994.

[19] Information on: http://arxiv.org/abs/1907.0218.